International Journal of Computer Science Engineering
and Information Technology Research (IJCSEITR)
ISSN(P): 2249-6831; ISSN(E): 2249-7943
Vol. 4, Issue 3, Jun 2014, 187-198


TRANS STELLAR
• Journal Publications • Research Consultancy

# CRYPTANALYTIC JH AND BLAKE HASH FUNCTION FOR AUTHENTICATION AND PROPOSED WORK OVER BLAKE-512 ON C LANGUAGE

**PRIYANKA WADHWANI, AKANKSHA GAUR & VIPIN JAIN**

Department of Computer Engineering, Swami Keshwanand Institute of Technology, Rajasthan Technical University, Jaipur, Rajasthan, India

## ABSTRACT

The paper aims to showcase work on improvement over SHA-512 with proposed work BLAKE-512 hash function. Hash functions form an important category of cryptography, which is widely used in a great number of protocols and security mechanisms. It is defined as computationally efficient function, which maps binary strings of arbitrary length to binary strings of fixed length. The last ones are the outputs of a hash computation and they are called hash values. Hash functions are applied to support digital signatures, data integrity, random number generators, authentication schemes, and data integrity mechanisms. National Institute of Standard and technology (NIST) focus on the new SHA-3 competition, started by the NIST, which searches for a new hash function in response to authentication concerns regarding the previous hash functions SHA-1 and the SHA-2 family. The paper aims to show that since SHA-512 used 80 rounds to calculate 512 bit final hash value with 512 bit initial hash value whereas BLAKE-512 is used with 16 rounds to calculate final hash value with same input length and same output length in C programming language.

**KEYWORDS:** BLAKE-512, NIST, SHA-3, SHA-512

## INTRODUCTION

Hash functions form an important category of cryptography, which is widely used in a great number of protocols and security mechanisms. Hash functions are mostly used to accelerate table lookup or data comparison tasks such as finding items in a database, detecting duplicate or similar records in a large file, finding similar stretches in DNA sequences, and so on.

### Hash Function BLAKE Consists of Four Steps

**Step 1: Padding the Message:** For BLAKE-256 the message is extended so that its length is congruent to 447 modulo 512. Length extension is performed by appending a bit 1 followed by a sufficient number of 0 bits. For BLAKE-512, message padding goes as follows: append a bit 1 and as many 0 bits until the message bit length is congruent to 895 modulo 1024. In the BLAKE-224 padded data, the 1 bit preceding the message length is replaced by a 0 bit. In the BLAKE-384 padded data, the preceding the message length is replaced by 0 bit.

m←m || 1000…0000 (l) 64 BLAKE-28

m←m || 1000…0001 (l) 64 BLAKE-32

m←m || 1000…0000 (l) 128 BLAKE-48

m←m || 1000…0001 (l) 128 BLAKE-64

**Step 2: Parsing the Padded Message:** In BLAKE-256, padded message is split into 16-word blocks $m^0$, . . ., $m^{N-1}$. We let $l^i$ be the number of message bits in $m^0$, . . .., $m^i$, that is, excluding the bits added by the padding. In BLAKE-224, the output is truncated to its first 224 bits. In BLAKE-384, the output is truncated to its first 384 bits. In BLAKE-256, the output is truncated to its first 256 bits.

Setting the initial hash value: BLAKE-512 uses the initial values of SHA-512. BLAKE-224 uses the eight initial values of SHA-224. BLAKE-384 uses the initial value of SHA-384. BLAKE-256 uses the eight initial values of SHA-256.

**Step 3: Message Digest:** In BLAKE-256 the iterated hash returns $h_0^N$,....., $h_7^N$. In BLAKE-224 iterated hash returns $h_0^N$,....., $h_6^N$ In BLAKE-384 iterated hash returns $h_0^N$,....., $h_5^N$.

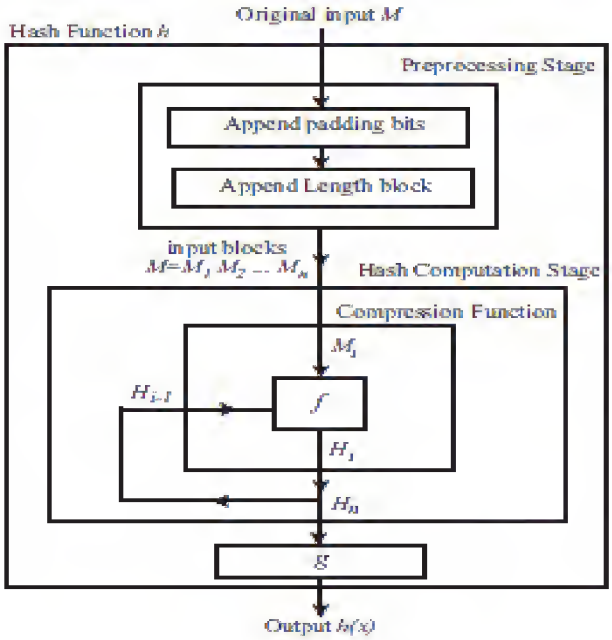


**Figure 1**

BLAKE-512 operates on 64-bit words and returns a 64-byte hash value. All lengths of variables are doubled compared to BLAKE-256: chain values are 512-bit, message blocks are 1024-bit, state is 256-bit, counter is 128-bit.

**Step 4: Compression Function:** The compression function of BLAKE-512 is similar to that of BLAKE-256 except that it makes 16 rounds instead of 14.

- It uses the initial values of SHA-512.
- BLAKE-32 and BLAKE-64 uses all outputs (256-bit & 512-bit each one) bits.
- Hashing the message
- For BLAKE-512, message padding goes as follows: append a bit 1 and as many 0 bits until the message bit length is congruent to 895 modulo 1024. Then append a bit 1, and a 128-bit unsigned big-endian representation of the message bit length:-

  $m \leftarrow m \| 1000.....0001 \| <l>_{128}$

## PROPOSED WORK

**BLAKE-512:** According to previous work of SHA-512, it uses the 80 rounds to generate the final hash value ($h^{'}$) with 1024 bit block size, 512 bit digest, 64 bit word length, 128 bit message length and 80 rounds but BLAKE-512 uses 16 rounds to generate final hash value ($h^{'}$) with same block size, digest, word length and message length. The initial hash value $H^{(0)}$ is the following sequence of 64-bit words (which are obtained by taking the fractional parts of the square roots of the first eight primes):

Initial Values

$IV_0$= 6A09E667 $IV_1$= BB67AE85

$IV_2$= 3C6EF372 $IV_3$= A54FF53A

$IV_4$= 510E527F $IV_5$= 9B05688C

$IV_6$= 1F83D9AB $IV_7$=5BE0CD19

**Table 1: Properties of SHA Hash Functions**

| Algorithm | Output size (bits) | Internal state size | Block size | Length size | Word size | Rounds |
|---|---|---|---|---|---|---|
| SHA-0 | 160 | 160 | 512 | 64 | 32 | 80 |
| SHA-1 | 160 | 160 | 512 | 64 | 32 | 80 |
| SHA-256/224 | 256/224 | 256 | 512 | 64 | 32 | 64 |
| SHA-512/384 | 512/384 | 512 | 1024 | 128 | 64 | 80 |

**Table 2: Properties of BLAKE Hash Functions (Sizes in Bits)**

| Algorithm | Word | Message | Block | Digest | Salt | Rounds |
|---|---|---|---|---|---|---|
| BLAKE-224 | 32 | $2^{64}$ | 512 | 224 | 128 | 14 |
| BLAKE-256 | 32 | $2^{64}$ | 512 | 256 | 128 | 14 |
| BLAKE-384 | 64 | $2^{128}$ | 1024 | 384 | 256 | 16 |
| BLAKE-512 | 64 | $2^{128}$ | 1024 | 512 | 256 | 16 |

**Table 3: Rounds of SHA-512 and BLAKE-512**

| Algorithm | SHA- 512 | BLAKE- 512 |
|---|---|---|
| Block size | 1024 | 1024 |
| Digest | 512 | 512 |
| Word | 64 | 64 |
| Length | 128 | 128 |
| **Rounds** | 80 | 16 |

Table 3 shows the 64 rounds is used in SHA-256 and 14 rounds is used in BLAKE-256, other properties are same for both candidates.

**BLAKE-512**

We have proposed three functions for calculation of final hash value. These three functions are following:

**Initialization**

This function uses the initial values of SHA-512.

**Initial Values**

The hash function BLAKE-512 operates on 64-bit words and returns a 32-byte hash value. This part defines BLAKE-512, going from its constant parameters to its compression function, then to its iteration mode.

BLAKE-512 starts hashing from the same initial value as SHA-512:

$IV_0$= 6A09E667 $IV_1$= BB67AE85

$IV_2$= 3C6EF372 $IV_3$= A54FF53A

$IV_4$= 510E527F $IV_5$= 9B05688C

$IV_6$= 1F83D9AB $IV_7$= 5BE0CD19

A 16-word state $v_0 \ldots v_{15}$is initialized such that different inputs produce different initial states. The state is represented as a 4×4 matrix shows in figure 6.1.

$$\begin{pmatrix} v_0 & v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 & v_7 \\ v_8 & v_9 & v_{10} & v_{11} \\ v_{12} & v_{13} & v_{14} & v_{15} \end{pmatrix} \leftarrow \begin{pmatrix} h_0 & h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 & h_7 \\ s_0 \oplus c_0 & s_1 \oplus c_1 & s_2 \oplus c_2 & s_3 \oplus c_3 \\ t_0 \oplus c_4 & t_0 \oplus c_5 & t_1 \oplus c_6 & t_1 \oplus c_7 \end{pmatrix}$$

**Figure 2: 4 x 4 Matrix**

In the initialization stage a large inner state is initialized from the previous chain value (h), salt (s), and counter (t). For the implementation of $v_9,\ldots, v_{15}$ a XOR chain is used, between the values of salt ($s_0,\ldots,s_3$) and the constant c ($c_0,\ldots,c_7$).

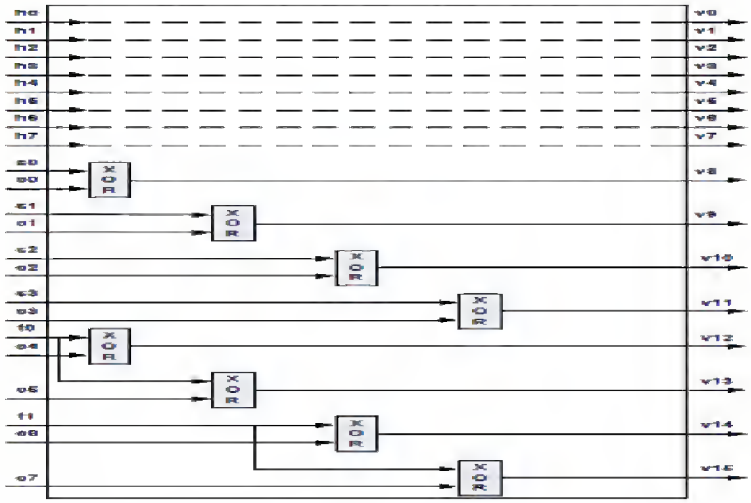

**Figure 3: Blake-64 Initialization Unit Architecture**

**Round Function**

The state value is then updated by message-dependent rounds.

Once the state v is initialized, the compression function iterates a series of 16 rounds. It is finally compressed to create the next chain value. In every round, the state v is transformed based on addition, XOR and right rotation computations, which are the components of $G_i$ (i=0,…,7) functions. For this purpose the $G_i$ functions are used. When the round sequence is taken over, the new chain value $h' = h'_0, \ldots, h'_7$ is produced from the state v, with inputs of the initial chain value $h_0, \ldots, h_7$ and the salt $s = s_0, \ldots, s_3$. Table 4 shows the permutation table which is used for solving $m_{\sigma r}$ and $c_{\sigma r}$ in below equations (1-8):

**Table 4: Permutation Table**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\sigma_0$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| $\sigma_1$ | 14 | 10 | 4 | 8 | 9 | 15 | 13 | 6 | 1 | 12 | 0 | 2 | 11 | 7 | 5 | 3 |
| $\sigma_2$ | 11 | 8 | 12 | 0 | 5 | 2 | 15 | 13 | 10 | 14 | 3 | 6 | 7 | 1 | 9 | 4 |
| $\sigma_3$ | 7 | 9 | 3 | 1 | 13 | 12 | 11 | 14 | 2 | 6 | 5 | 10 | 4 | 0 | 15 | 8 |
| $\sigma_4$ | 9 | 0 | 5 | 7 | 2 | 4 | 10 | 15 | 14 | 1 | 11 | 12 | 6 | 8 | 3 | 13 |
| $\sigma_5$ | 2 | 12 | 6 | 10 | 0 | 11 | 8 | 3 | 4 | 13 | 7 | 5 | 15 | 14 | 1 | 9 |
| $\sigma_6$ | 12 | 5 | 1 | 15 | 14 | 13 | 4 | 10 | 0 | 7 | 6 | 3 | 9 | 2 | 8 | 11 |
| $\sigma_7$ | 13 | 11 | 7 | 14 | 12 | 1 | 3 | 9 | 5 | 0 | 15 | 4 | 8 | 6 | 2 | 10 |
| $\sigma_8$ | 6 | 15 | 14 | 9 | 11 | 3 | 0 | 8 | 12 | 2 | 13 | 7 | 1 | 4 | 10 | 5 |
| $\sigma_9$ | 10 | 2 | 8 | 4 | 7 | 6 | 1 | 5 | 15 | 11 | 9 | 14 | 3 | 12 | 13 | 0 |

Each round of transformation is made of eight operations using the $G_i$ functions defined as follows or a round is a transformation of the state v that computes as.

$G_0(v_0, v_4, v_8, v_{12})$ $G_1(v_1, v_5, v_9, v_{13})$ $G_2(v_2, v_6, v_{10}, v_{14})$ $G_3(v_3, v_7, v_{11}, v_{15})$

$G_4(v_0, v_5, v_{10}, v_{15})$ $G_5(v_1, v_6, v_{11}, v_{12})$ $G_6(v_2, v_7, v_8, v_{13})$ $G_7(v_3, v_4, v_9, v_{14})$

where, at round r, $G_i(a, b, c, d)$ sets:

$a \leftarrow a + b + (m_{\sigma r}(2i)\ \text{XOR} c_{\sigma r}(2i+1))$ 1

$d \leftarrow (d\ \text{XOR} a) >>> 16$ 2

$c \leftarrow (c + d)$ 3

$b \leftarrow (b \text{ XORc}) >>> 12$ 4

$a \leftarrow a + b + (m_{\sigma r}(2i+1) \text{ XORc}_{\sigma r}(2i))$ 5

$d \leftarrow (d \text{ XORa}) >>> 8$ 6

$c \leftarrow (c + d)$ 7

$b \leftarrow (b \text{ XORc}) >>> 7$ 8

From the above equations (1-8) is obvious that for each one of the variables a,…, d two district values are set during one round transformation [1]. For each one of $G_i$ functions an alternative implementation is proposed. All of them could be achieved based on the same generic $G_i$ architecture, which is basically consists of n-bit modulo adders, XOR chains and, shift operations and registers. In order to implement the $G_i$ functions in below figure 6.3

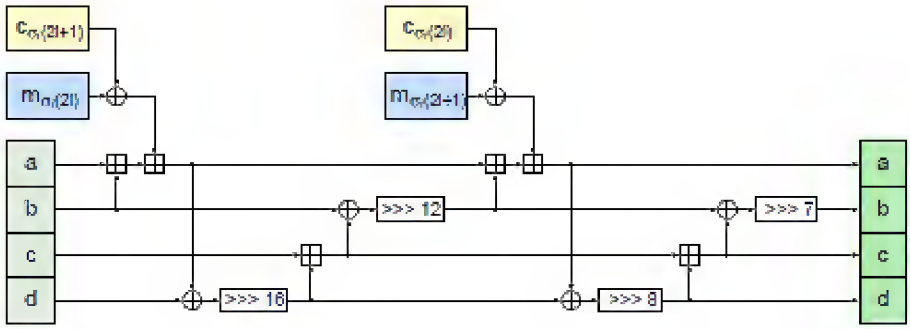


**Figure 4: Gifunction Structure**

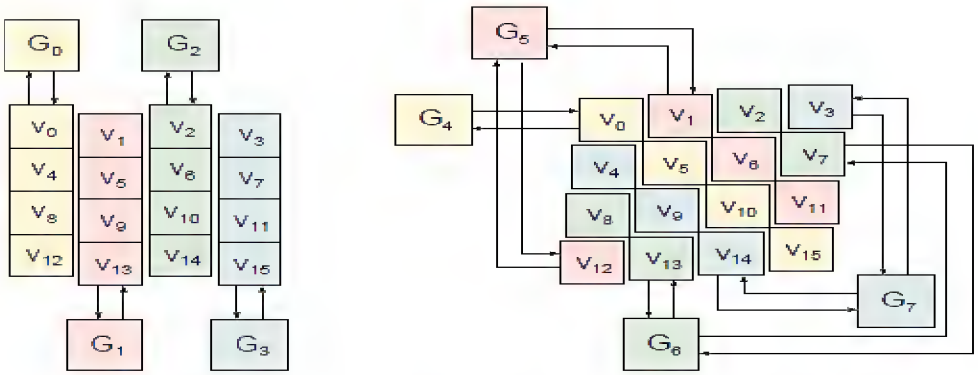


**Figure 5: $G_i$ Functions Transformation: (a) Column Step (b) Diagonal Step**

In figure 6.4 shows the first four calls $G_{0,\ldots,}$ $G_3$ can be computed in parallel. We call the procedure of computing $G_{0,\ldots}$, $G_3$ a column step. The last four calls $G_{4,}$. . ., G7 update distinct diagonals thus can be parallelized as well, which we call a diagonal step. At round r > 9, the permutation used is $\sigma_{r \bmod 10}$.

**Finalization**

After the rounds sequence, the new chain value $h'_0, \ldots, h'_7$is extracted from the state $v_0, \ldots, v_{15}$with input of the initial chain value $h_0, \ldots, h_7$and the salt $s_0, \ldots, s_3$as follows [3]:

$h'_{0} \leftarrow h_0 XOR s_0 \text{ XOR } v_0 XOR v_8$

$h'_{1} \leftarrow h_1 XOR s_1 \text{ XOR } v_1 XOR v_9$

$h'_{2} \leftarrow h_2 XOR s_2 \text{ XOR } v_2 XOR v_{10}$

$h'_{3} \leftarrow h_3 XOR s_3 \text{ XOR } v_3 XOR v_{11}$

$h'_{4} \leftarrow h_4 XOR s_0 \text{ XOR } v_4 XOR v_{12}$

$h'_{5} \leftarrow h_5 XOR s_1 \text{ XOR } v_5 XOR v_{13}$

$h'_{6} \leftarrow h_6 XOR s_2 \text{ XOR } v_6 XOR v_{14}$

$h'_{7} \leftarrow h_7 XOR s_3 \text{ XOR } v_7 XOR v_{15}$

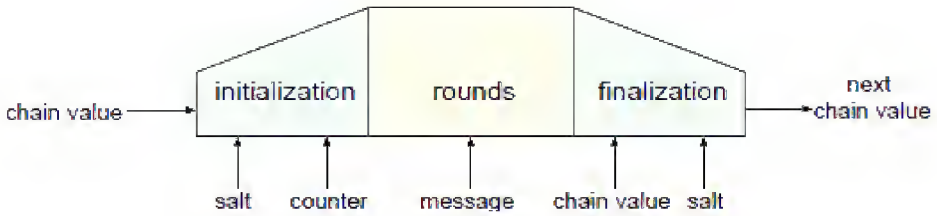


**Figure 6: The Local Wide-Pipe Construction of Blake's Compression Function**

It has been estimated that five rounds is the minimum number that must be executed for BLAKE -28 & - 32 bit, although ten rounds are recommended. For BLAKE-48, -64 fourteen rounds are recommended, although the functions could operate sufficiently with seven rounds of data transformations.

## EVALUATION

We have used simulation tool (Borland Turbo C) to show simulation work: It gives us C code computing the compression function of BLAKE-512 with Header File BLAKEh.h and Turbo file Turbo.c. Firstly, It calculates $G_0$ ($v_0$, $v_4$, $v_8$, $v_{12}$)

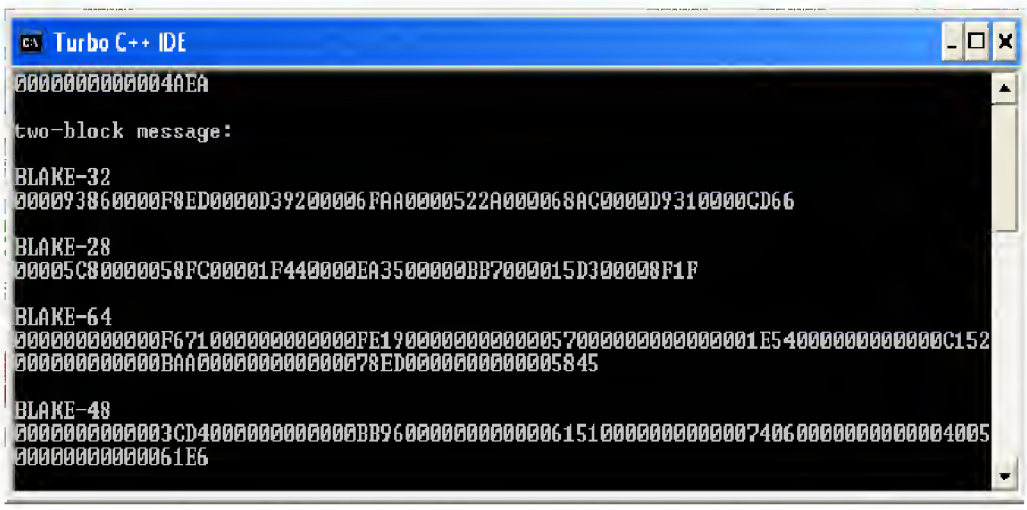


**Figure 7: Hash Values Obtained after Two Block Message**

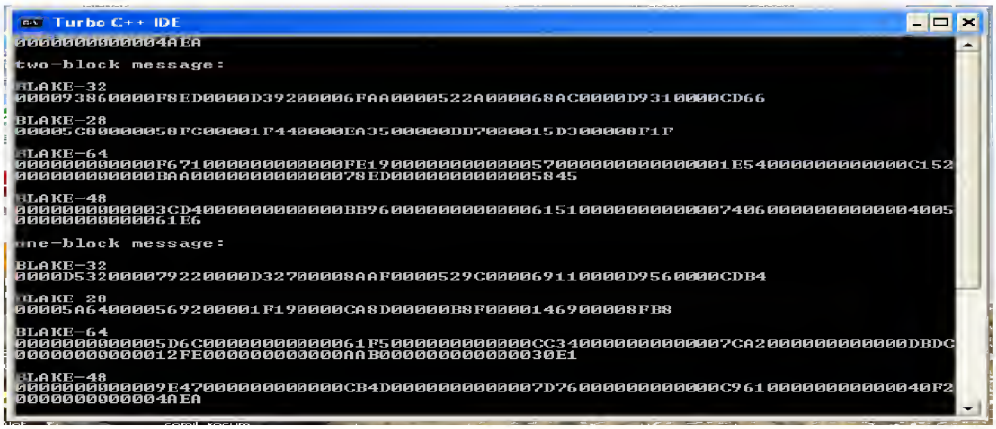


**Figure 8: Hash Value Obtained after One-Block Message**

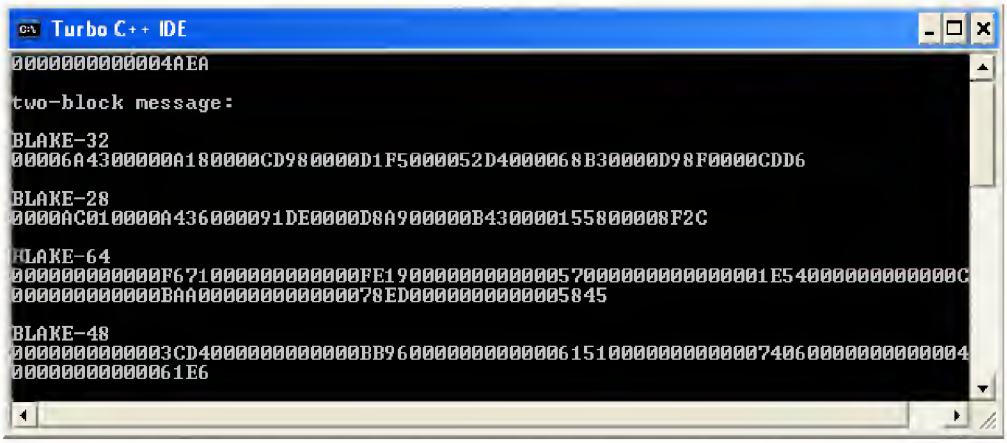


**Figure 9: Changed Hash Value Obtained after Two-Block Message**

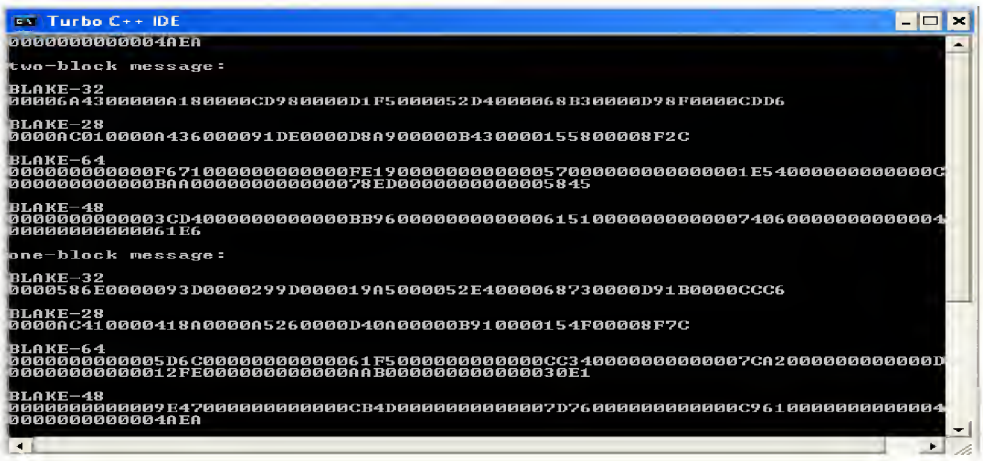


**Figure 10: Changed Hash Value Obtained after One-Block Message**

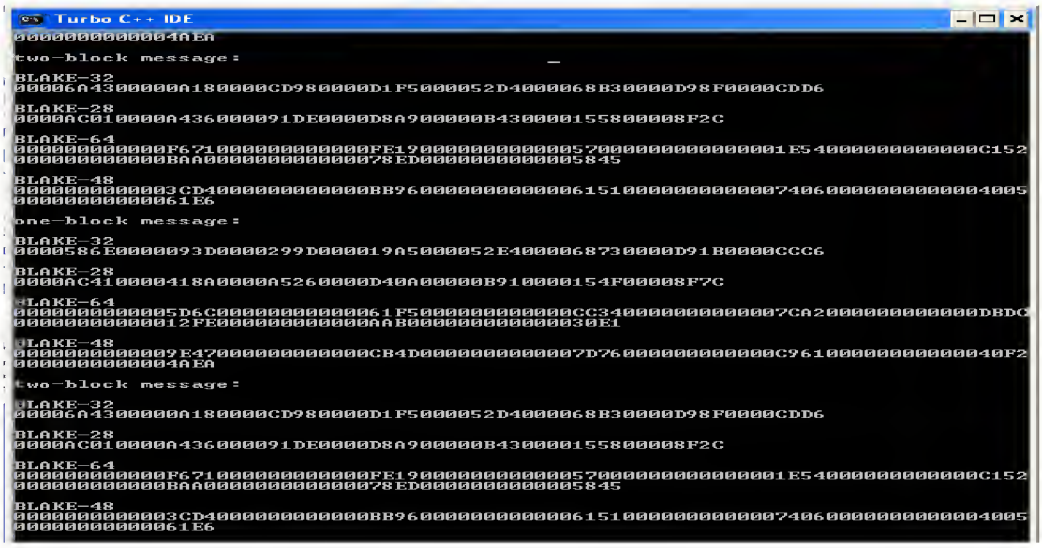


**Figure 11: Changed Combined Hash Value of One & Two Block-Message**

## CONCLUSIONS

A cryptographic hash function is a hash function, that is, an algorithm that takes an arbitrary block of data and returns a fixed-size bit string, the (cryptographic) hash value, such that an (accidental or intentional) change to the data will (with very high probability) change the hash value. In this work the comparison of two SHA-3 cryptography hash function JH and BLAKE is done, which is less time consuming process. BLAKE-256 with 14 rounds instead of 64 rounds of SHA-256 is more reliable in comparison to SHA family. This work gives the comparative hash algorithms of both candidates.

## FUTURE WORK

In future the working of the function (BLAKE-512) discussed here should be more less time consuming. It can merge all three function of BLAKE-512 into single unit then give one time input (512 bit) and get 512 bit final hash value using xilinx.

## ACKNOWLEDGEMENTS

Through this page, I express my heartfelt thanks, to Mr. Vipin Jain (Senr. Lect.), Computer Science Department at Swami Keshvanand Institute of Technology, Management and Gramothan Jaipur, who gave me the opportunity to work on this topic and inspired me to carry forward this work as a challenge.

## REFERENCES

1. Bernhard Jungk and J¨urgenApfelbeck, HochschuleRheinMain, Wiesbaden, Germany ……2011 International Conference on Reconfigurable Computing
2. Cadence Design Systems. The Cadence Design Systems Website.http://www.cadence.com/.
3. Canavan, John E. Fundamentals of network security / John E. Canavan. p. cm.—(Artech House telecommunications library) Includes bibliographical references and index. ISBN1-58053-176-8 (alk. paper)http://www.artechhouse.com

4. D. J. Bernstein. ChaCha, a variant of Salsa20. Available online at http://cr.yp.....to/chacha/ chacha-20080128.pdf, January 2008.
5. Dr. Rahul Banerji Computer science and information security group "Introduction to Network Security".
6. E. Biham, O. Dunkelman, "A framework for iterative hash functions- HAIFA". Cryptologye Print Archive, Report 2007/278.
7. E. Biham and O. Dunkelman. A Framework for Iterative Hash Functions - HAIFA. In .....Second NIST Cryptographic Hash Workshop, Santa Barbara, California, USA, August .....24-25, 2006, August 2006.
8. George Provelengios, Nikolaos S. Voros, ParisKitsos Greece, 2011 14th Euromicro. Conference on Digital System Design
9. H. Namin and M. A. Hasan, Waterloo, Ontario N2L 3G1 Canada. Compression ..... Function for Selected SHA-3 Candidates.
10. Hongjun Wu, "The Hash Function JH", The First SHA-3 Candidate Conference, 2009, available on line at http://ehash.iaik.tugraz.at/wiki/JH
11. H. Wu. SHA-3 proposal JH, version January 15, 2009. JH
12. http://en.wikipedia.org/wiki/secure_hash_algorithms
13. http://www.iwar.org.uk/comsec.resources/cipher/sha256-384-512.pdf
14. IEEE Annual Symposium Nicolas Sklavos 2010, ZIP 27100, GREECE, Paris Kitsos, .....Computer Science Hellenic Open University, GREEC.
15. Iterative Difierentials, Symmetries, and Message Modification in BLAKE-256 Orr .....Dunkelman and Dmitry Khovratovich University of Haifa, Israel Weizmann. University, Israel Microsoft Research Redmond, USA
16. J.-L. Beuchat, E. Okamoto, and T. Yamazaki, "Compact implementations of BLAKE-.....32 and BLAKE-64 on FPGA," in FPT, 2010, pp. 170–177.
17. J. P. Aumasson, L. Henzen, W. Meier, and R.C.W. Phan, "SHA-3 Proposal BLAKE", online: http://www.131002.net/blake 2010.
18. J.P. Aumasson, L. Henzen, W. Meier, R.C.W. Phan, "SHA-3 proposal BLAKE". Submission to the SHA-3 Competition, 2008
19. Mao Ming, HeQiang, Shaokun Zeng Xidian University Xi'an, Shanxi, China BLAKE-..32 based on differential properties, 2010 International Conference on Computational and Information Sciences.
20. National Institute of Standard and Technology (NIST): Cryptographic Hash Algorithm. Competition Website: http://csrc.nist.gov/groups/ST/hash/sha-3/.
21. Secure Hash Standard (SHS), National Institute of Standards and Technology.....(NIST), FIPS. PUB..180-3, 2008, available on line at http://csrc.nist.gov/publications/fips/fips1803/fips1803_final.pdf

22. SHA-1 Standard, National Institute of Standards and Technology (NIST), Secure Hash ……Standard, FIPS PUB 180-1, 1995, available on line at www.itl.nist.gov/fipspubs/ fip180 ……1.htm

23. SHA-512/256 Shay Gueron 1, 2, Simon Johnson 3, Jesse Walker4 Department of ……Mathematics, University of Haifa, Israel Mobility Group, Intel Corporation, Israel ……Development Center, Haifa, Israel Intel Architecture Group, Intel Corporation, USA ……Security Research Lab, Intel Labs, Intel Corporation, USA

24. Wu, H., SHA-3 proposal JH, Website: http://icsd.i2r.astar. edu.sg/staff/hongjun/jh/.